UNIVERSITE MOHAMMED V RABAT-AGDAL

Faculté des Sciences

Département d'Informatique

SMI - Algo.II, 2014-2015

Série 3

EX.1

On considère un tableau T à n éléments contenant des 0, des 1 et des 2.

Ecrire un algorithme qui regroupe tous les 0 au début du tableau T, suivi du bloc des 2 et enfin le bloc des 1.

Ex.2

1) Ecrire un algorithme récursif qui calcule la somme de chiffres d'un entier n ($n \geq 0$).
2) La multiplication russe de deux entiers positifs a et b consiste à diviser a par 2 (et ensuite les quotients obtenus) jusqu'à ce qu'on arrive à un quotient nul, et à chaque division par 2, on multiplier b par 2. On additionne les multiples de b correspondant aux restes de la division non nuls.
   a) Ecrire une fonction récursive qui retourne la multiplication de a et b par cette méthode
   b) Donner une version itérative.

Ex.3

1) Donner un algorithme itératif qui retourne l'indice du maximum dans un tableau T à n entiers.
2) Ecrire un algorithme équivalent en récursif.
3) Utiliser la méthode DpR pour chercher l'indice du maximum.
4) Par la méthode de tournoi. On organise un tournoi en comparant les éléments deux par deux, les vainqueurs (les plus grands) seront en compétition à l'étape suivante, etc, jusqu'à ce qu'on aura un seul vainqueur.

Ex.4

Etant donné n soldats et deux enfants sur une rive d'une rivière. Il s'agit de faire traverser les n soldats à l'autre coté de la rivière à l'aide d'une barque ; cette barque ne peut prendre qu'une seule personne ou les deux enfants.

a) Expliquer comment faire passer les n soldats de l'autre coté de la rivière, en ramenant le problème de la traversée de n soldats à celle de n-1 soldats.
b) Calculer le nombre d'aller-retour que la barque doit effectuer.

Ex.5

On considère l'algorithme suivant :

```
Mystère(T, i, n)
début
  si  i < 1  alors  afficher(T, n)
  sinon
        T[i] :=  0 ;
        Mystère(T, i-1, n) ;
        T[i] :=  1 ;
        Mystère(T, i-1, n) ;
```

```
afficher(T, n)
début
   pour i :=1 à n faire
          écrire(T[i])
   fpour
écrireln() ;//retour à la ligne
fin
```

fsi

fin

1) Dresser l'arbre des appels de Mystère(T, 3, 3) et donner les différents affichages de T dans l'ordre de l'exécution de l'algorithme.
2) Dire, en quelques mots, ce que fait l'algorithme Mystère(T, n, n).
3) Evaluer sa complexité

Ex.6

La recherche séquentielle d'un élément dans un tableau à n éléments est de l'ordre de n, et la recherche dichotomique est de l'ordre de $\log_2 n$ pour un tableau trié.

Etant donné un tableau T à n éléments triés dans l'ordre croissant. On partage T en sous-tableaux de taille k chacun, sauf (peut être) pour le dernier, où $1 \leq k \leq n$ et $mk \leq n < (m+1)k$.

Pour chercher un élément x dans le tableau T, on compare x à T[k], puis à T[2k] et ainsi de suite. Si on trouve un p ($1 \leq p \leq m$) tel que $x \leq T[p*k]$, on fait une recherche séquentielle dans le sous-tableau p, sinon la recherche se fait dans le dernier sous-tableau jusqu'à la fin du tableau.

1) Ecrire un algorithme RechParSaut(T, inf, sup, k, x) pour chercher x dans le tableau T[inf..sup] utilisant la méthode décrite ci-haut. On retourne la position de x si x est dans le tableau, 0 sinon.
2) On note par f(n, k) le nombre de comparaisons de cet algorithme dans le pire des cas. Donner l'expression de f(n, k).
3) Trouver la (ou les) valeur(s) de k qui minimise (nt) le nombre de comparaisons f(n, k).
4) Comparer cet algorithme de recherche avec la recherche dichotomique.

Ex.7

Soit un immeuble à n étages numérotés de 1 à n. On dispose de k étudiants.

Il s'agit de faire sauter un étudiant par la fenêtre d'un étage de l'immeuble pour déterminer à partir de quel étage le saut est fatal.

Proposer une méthode, pour chacun des cas suivants, qui cherche la hauteur à partir de laquelle un saut est fatal en faisant un nombre minimum de sauts :

a) k= 1
b) k = 2
c) $k \geq \log_2 n$
d) $k < \log_2 n$

(si un étudiant survit à un saut, on peut le réutiliser pour les sauts suivants)

EX.8

Soit A un tableau de n entiers. Un élément de A est dit majoritaire s'il a une fréquence supérieure à n/2.

1) Ecrire un algorithme qui vérifie si A contient un élément majoritaire. Calculer sa complexité.
2) Proposer un algorithme qui utilise la méthode D.p.R pour la recherche d'un élément majoritaire de A. L'algorithme retourne le couple (vrai, x) si x est majoritaire dans A et il retournera (faux, 0) si A n'a pas d'élément majoritaire.